HOMEWORK - SUMMER 2016

HOMEWORK 1 - due Tuesday, July 19th no later than 7:00pm

REMINDERS:

- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework assignments MUST be entirely your own work. Also, group efforts are not allowed.***
- **Login to your grading account and click "Submit Assignment" to upload and submit your assignment.**
- **You are not allowed to use ArrayList, Vector or any other Java API Data Structure classes to implement this assignment.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

In this assignment, you will be required to write an ADT that represents a playlist of audio files similar to the ones used by audio playback programs such as iTunes and Windows Media Player. Your ADT will be responsible for maintaining simple information about each audio file (namely the title, artist, and length in minutes and seconds) as well as the correct ordering of the audio files in the list. The ADT should also provide a series of operations for manipulating the playlist including adding and removing audio files, displaying the contents of the list, and finding all songs that were performed by a particular artist.

1. Write a fully-documented class named SongRecord which contains information about a particular audio file. It should have member variables for the title and artist (both strings) as well as two member variables for the song's length in minutes and seconds (both ints). You should provide accessor and mutator methods for each variable as well as a default constructor. For the mutator method of the seconds variable, you should throw an exception if the new value is less than 0 or greater than 59. For the mutator method of the minutes variable, you should throw an exception if the new value is negative. Finally, you should provide a toString() method that neatly prints the information about the audio file on a single line as shown below in the sample output.

2. Write a fully-documented class named Playlist that stores all SongRecord objects that belong to a particular playlist. The SongRecord objects should be stored in an array. There should be a maximum of 50 SongRecord objects allowed, a number which should be defined as a final variable. The class will be based on the following ADT specification:

> `public class Playlist`
> **The Playlist class implements an abstract data type for a playlist of audio files supporting common operations on such lists of audio files.**

- **Constructor for Playlist**
  `public Playlist()`
  Construct an instance of the Playlist class with no SongRecord objects in it.

Postcondition:
This Playlist has been initialized to an empty list of SongRecords.

- **clone**
  `public Object clone()`
  Generate a copy of this Playlist.
  Returns:
  The return value is a copy of this Playlist. Subsequent changes to the copy will not affect the original, nor vice versa. Note that the return value must be typecast to a Playlist before it can be used.
- **equals**
  `public boolean equals (Object obj)`
  Compare this Playlist to another object for equality.
  Parameters:
  `obj` **- an object in which this Playlist is compared**
  Returns:
  A return value of true indicates that `obj` refers to a Playlist object with the same SongRecords in the same order as this Playlist. Otherwise, the return value is false.
  Note:
  If `obj` is null or it is not a Playlist object, then the return value is false.
  Note:
  When comparing equality between two SongRecord objects, you must verify that their titles, artists, and song lengths are all the same. Using the == operator will simply check to see if the two variables *refer to the same SongRecord object,* which does not take into consideration that two different SongRecord objects can actually represent the same audio file. To solve this problem, you can either check that each of the properties of the two objects are the same (title, artist, and length) inside of this method, or you may simplify this process by implementing an equals method (similar to this one) for the SongRecord class.
- **size**
  `public int size()`
  Determines the number of SongRecords currently in this Playlist.
  Preconditions:
  This SongRecord object has been instantiated.
  Returns:
  The number of SongRecords in this Playlist.
- **addSong**
  `public void addSong(SongRecord song, int position)`
  Parameters:
  `song` **- the new SongRecord object to add to this Playlist**
  `position` **- the position in the playlist where the `song` will be inserted**
  Preconditions:
  This SongRecord object has been instantiated and 1 ≤ `position` ≤ songs_currently_in_playlist + 1. The number of SongRecord objects in this Playlist is less than max_songs.
  Postcondition:
  The new SongRecord is now stored at the desired position in the Playlist. All SongRecords that were originally in positions greater than or equal to position are moved back one position. (Ex: If there are 5 songs in a Playlist, positions 1-5, and you insert a new SongRecord at position 4, the new SongRecord will now be at position 4, the SongRecord that was at position 4 will be moved to position 5, and the SongRecord that was at position 5 will be moved to position 6).
  Throws:
  `IllegalArgumentException`
  Indicates that position is not within the valid range.
  `FullPlaylistException`

Indicates that there is no more room inside of the Playlist to store the new SongRecord object.
Note 1:
`position` **refers to the position in the Playlist and not the position inside the array.**
Note 2:
Inserting a song to position (songs_currently_in_playlist + 1) is effectively the same as adding a song to the end of the Playlist.

- **removeSong**
  `public void removeSong(int position)`
  Parameters:
  `position` **- the position in the playlist where the song will be removed from.**
  Preconditions:
  This SongRecord object has been instantiated and 1 ≤ `position` **≤ songs_currently_in_playlist.**
  Postcondition:
  The SongRecord at the desired position in the Playlist has been removed. All SongRecords that were originally in positions greater than or equal to position are moved forward one position. (Ex: If there are 5 songs in a Playlist, positions 1-5, and you remove the SongRecord at position 4, the SongRecord that was at position 5 will be moved to position 4).
  Throws:
  `IllegalArgumentException`
  Indicates that `position` **is not within the valid range.**
  Note:
  `position` **refers to the position in the Playlist and not the position inside the array.**
- **getSong**
  `public SongRecord getSong(int position)`
  Get the SongRecord at the given position in this Playlist object.
  Parameters:
  `position` **- position of the SongRecord to retrieve**
  Preconditions:
  This Playlist object has been instantiated and 1 ≤ `position` **≤ songs_currently_in_playlist.**
  Returns:
  The SongRecord at the specified position in this Playlist object.
  Throws:
  `IllegalArgumentException`
  Indicates that position is not within the valid range.
  Note:
  `position` **refers to the position in the Playlist and not the position inside the array.**
- **printAllSongs**
  `public void printAllSongs()`
  Prints a neatly formatted table of each SongRecord in the Playlist on its own line with its position number as shown in the sample output.
  Preconditions:
  This SongRecord object has been instantiated.
  Postcondition:
  A neatly formatted table of each SongRecord in the Playlist on its own line with its position number has been displayed to the user.
  Note:
  `position` **refers to the position in the Playlist and not the position inside the array.**
  Hint:
  If your toString() method is implemented correctly as described below, you will simply need to call it and print the results to the user.
- **getSongsByArtist**

`public static Playlist getSongsByArtist(Playlist originalList, String artist)`
Generates a new Playlist containing all SongRecords in the original Playlist performed by the specified artist.
Parameters:
`originalList` **- the original Playlist**
`artist` **- the name of the artist**
Preconditions:
The Playlist referred to by originalList has been instantiated.
Returns:
A new Playlist object containing all SongRecords in the original Playlist performed by the specified artist.
Note:
The return value is null if either `originalList` or `artist` is null.
Note:
The order of the SongRecords in the new Playlist should relate to the order of the SongRecords in the old Playlist. For example, if the original Playlist has 8 SongRecords, positions 1-8, and SongRecords 3, 6, and 8 were performed by the specified artist, the new Playlist should have the SongRecord originally at position 3 placed at location 1, the SongRecord originally at position 6 placed at location 2, and the SongRecord originally at position 8 placed at location 3.

- **toString**
  public String toString()
  Gets the String representation of this Playlist object, which is a neatly formatted table of each SongRecord in the Playlist on its own line with its position number as shown in the sample output.
  Returns:
  The String representation of this Playlist object.
  Note:
  Position refers to the position in the Playlist and not the position inside the array.

3. Write a fully documented class named PlaylistOperations that is based on the following specification:

public class PlaylistOperations
The PlaylistOperations Java application tests the methods of the Playlist class and allows the user to manipulate a single Playlist object by performing operations on it.

- **main public static void main(String[] args)**
  The main method runs a menu driven application which first creates an empty Playlist and then prompts the user for a menu command selecting the operation. The required information is then requested from the user based on the selected operation. Following is the list of menu options and their required information:

```
Add Song:                A  <Title> <Artist> <Minutes> <Seconds> <Position>
Get Song:                G  <Position>
Remove Song:             R  <Position>
Print All Songs:         P
Print Songs By Artist:   B  <Artist>
Size:                    S
Quit:                    Q
```

4. You will also need a class to handle the exception FullPlaylistException.

Note: You may include additional methods in the SongRecord, Playlist, or PlaylistOperations as

necessary.

INPUT FORMAT:

- Each menu operation is entered on its own line and should be case insensitive (i.e. 'q' and 'Q' are the same).
- Check to make sure that the position, if required, is valid. If not, print an error message and return to the menu.
- For the Add Song command, if the input information is valid, construct the object accordingly. Otherwise, print an error message and return to the menu.
- You may assume that the lengths of the input for the song titles and artists are less than 25 characters long.

OUTPUT FORMAT:

- Echo the input information for the Add Song command in the output.
- All menu operations must be accompanied by a message indicating what operation was performed and whether or not it was successful.
- The seconds of the song length must always be printed as two digits (Ex: 3:09 is valid but 3:9 is not).
- All lists must be printed in a nice and tabular form as shown in the sample output. You may use C style formatting as shown in the following example. The example below shows two different ways of displaying the name and address at pre-specified positions 21, 26, 19, and 6 spaces wide. If the '-' flag is given, then it will be left-justified (padding will be on the right), else the region is right-justified. The 's' identifier is for strings, the 'd' identifier is for integers. Giving the additional '0' flag pads an integer with additional zeroes in front.

```
    String name = "Doe Jane";
    String address = "32 Bayview Dr.";
    String city = "Fishers Island, NY";
    int zip = 6390;

System.out.println(String.format("%-21s%-26s%19s%06d", name, address, city, zip));
System.out.printf("%-21s%-26s%19s%06d", name, address, city, zip);

Doe Jane              32 Bayview Dr.              Fishers Island, NY 06390
Doe Jane              32 Bayview Dr.              Fishers Island, NY 06390
```

HINTS:

- Remember that the position parameter to all of the methods listed in the Playlist class refers to the song at a given position within a playlist (starting at position 1) and not the position inside of the array (which starts at position 0). There are two ways that you can handle this issue:
  - Store song 1 in array position 0, song 2 in array position 1, and so on and so forth. Inside each method, subtract one from the position given by the parameter to find the appropriate position within the array.
  - Define your array such that it is of size MAX_SONGS + 1 instead of MAX_SONGS. Store song 1 in array position 1, song 2 in array position 2, and so on and so forth. Position 0 of the array will not be used.

EXTRA CREDIT:

- Provide an option to play the song. For this option, you should input the song's filename along with other information related to each song (title, artist, duration, etc.). [10 points]
- Use GUI for all the user interface (input/output). [up to 13 points depending on how well you incorporate the GUI options]

- **Add the following menu options to create and manage multiple playlists: [15 points]**
  1. **N - Create a new playlist and set as current playlist. Input the playlist name from the user.**
  2. **V - Change current playlist. Input the playlist name from the user.**
  3. **C - Copy the current playlist's songs into a new playlist. Input the new playlist name from the user.**
  4. **E - Compare the songs in the current playlist with the given playlist. Input the given playlist name from the user.**
  5. **D - Display all playlist names.**

SAMPLE INPUT/OUTPUT:

Note: User input is in black, computer generated output appears in blue and comments are in green.

```
A) Add Song
B) Print Songs by Artist
G) Get Song
R) Remove Song
P) Print All Songs
S) Size
Q) Quit

Select a menu option: A

Enter the song title: Radioactive
Enter the song artist: Imagine Dragons
Enter the song length (minutes): 4
Enter the song length (seconds): 28
Enter the position: 1
Song Added: Radioactive By Imagine Dragons
```

```
// menu not shown in the sample input/output
Select a menu option: A

Enter the song title: Push
Enter the song artist: Matchbox 20
Enter the song length (minutes): 3
Enter the song length (seconds): 59
Enter the position: 1
Song Added: Push By Matchbox 20

// menu not shown in the sample input/output
Select a menu option: P


Song#     Title           Artist          Length
-----------------------------------------------
1         Push            Matchbox 20      3:59
2         Radioactive     Imagine Dragons  4:28


// menu not shown in the sample input/output
Select a menu option: A

Enter the song title: Gangnam Style
Enter the song artist: PSY
Enter the song length (minutes): 4
Enter the song length (seconds): 9
Enter the position: 2
Song Added: Gangnam Style By PSY

// menu not shown in the sample input/output
```

```
Select a menu option: P


Song#     Title           Artist          Length
-----------------------------------------------
1         Push            Matchbox 20     3:59
2         Gangnam Style   PSY             4:09
3         Radioactive     Imagine Dragons 4:28


Select a menu option: S
There are 3 song(s) in the current playlist.

// menu not shown in the sample input/output
Select a menu option: R

Enter the position: 1
Song Removed at position 1

// menu not shown in the sample input/output
Select a menu option: P


Song#     Title           Artist          Length
-----------------------------------------------
1         Gangnam Style   PSY              4:09
2         Radioactive     Imagine Dragons  4:28
```

```
// menu not shown in the sample input/output
Select a menu option: A

Enter the song title: It's Time
Enter the song artist: Imagine Dragons
Enter the song length (minutes): 5
Enter the song length (seconds): 24
Enter the position: 3
Song Added: It's Time By Imagine Dragons

// menu not shown in the sample input/output
Select a menu option: P


Song#     Title           Artist          Length
-----------------------------------------------
1         Gangnam Style   PSY              4:09
2         Radioactive     Imagine Dragons  4:28
3         It's Time       Imagine Dragons  5:24


// menu not shown in the sample input/output
Select a menu option: B

Enter the artist: Imagine Dragons


Song#     Title           Artist          Length
-----------------------------------------------
1         Radioactive    Imagine Dragons  4:28
2         It's Time      Imagine Dragons  5:24
```

```
//Note the song numbers in comparison with the previous example:

// menu not shown in the sample input/output
Select a menu option: P


Song#     Title           Artist          Length
-----------------------------------------------
1         Gangnam Style   PSY              4:09
2         Radioactive     Imagine Dragons  4:28
3         It's Time       Imagine Dragons  5:24



// menu not shown in the sample input/output
Select a menu option: G

Enter the position: 2

Song#     Title           Artist          Length
-----------------------------------------------
2         Radioactive     Imagine Dragons  4:28



//No songs found by artist:
Select a menu option: B
Enter the artist: Blink 182


Song#     Title           Artist          Length
-----------------------------------------------



//Invalid Input examples:
Select a menu option: R
Enter the position: 4
No song at position 4 to remove.

// menu not shown in the sample input/output
Select a menu option: A

Enter the song title: Some Other
Enter the song artist: Song
Enter the song length (minutes): 2
Enter the song length (seconds): 14
Enter the position: 9
Invalid position for adding the new song.

// menu not shown in the sample input/output
Select a menu option: A

Enter the song title: Some Other
Enter the song artist: Song
Enter the song length (minutes): 2
Enter the song length (seconds): 214
Enter the position: 3
Invalid song length.

// menu not shown in the sample input/output
Select a menu option: Q

Program terminating normally...
```